02

Criando funções para os nossos jogos

Transcrição

No vídeo anterior, vimos o problema do `import` acontecendo. Para resolver isso, vamos fazer algo semelhante à função `print()`, ela existe, mas só é executada quando chamada. Ou seja, vamos criar uma função específica para cada jogo.

Colocando o código dos jogos dentro de funções

Em Python, quando queremos criar uma função, precisamos defini-la, através da palavra chave `def`. Vamos começar definindo a função `jogar()` no arquivo forca.py:

```
# forca.py

def jogar():

    print("*********************************")
    print("***Bem vindo ao jogo da Forca!***")
    print("*********************************")

    print("Fim do jogo")
```

Faremos a mesma coisa no jogo de adivinhação, colocando todo o seu código dentro da função específica `jogar()`:

```
# adivinhacao.py

import random

def jogar():
    print("*********************************")
    print("Bem vindo ao jogo de Adivinhação!")
    print("*********************************")

    numero_secreto = random.randrange(1, 100)
    total_de_tentativas = 0
    pontos = 1000

    print("Qual o nível de dificuldade?")
    print("(1) Fácil (2) Médio (3) Difícil")

    nivel = int(input("Defina o nível: "))

    if (nivel == 1):
        total_de_tentativas = 20
    elif (nivel == 2):
        total_de_tentativas = 10
    else:
        total_de_tentativas = 5

    for rodada in range(1, total_de_tentativas + 1):
        print("Tentativa {} de {}".format(rodada, total_de_tentativas))
```

```
        chute_str = input("Digite um número entre 1 e 100: ")
        print("Você digitou: ", chute_str)
        chute = int(chute_str)

        if (chute < 1 or chute > 100):
            print("Você deve digitar um número entre 1 e 100!")
            continue

        acertou = numero_secreto == chute
        maior = chute > numero_secreto
        menor = chute < numero_secreto

        if (acertou):
            print("Você acertou e fez {} pontos!".format(pontos))
            break
        else:
            if (maior):
                print("Você errou! O seu chute foi maior que o número secreto.")
            elif (menor):
                print("Você errou! O seu chute foi menor que o número secreto.")
            pontos_perdidos = abs(numero_secreto - chute)
            pontos = pontos - pontos_perdidos

    print("Fim do jogo")
```

Chamando as funções

Agora precisamos chamar essas duas funções dentro do arquivo **jogos.py**, utilizando o nome do módulo e chamando a sua função:

```
# jogos.py

import forca
import adivinhacao

print("**********************************")
print("*******Escolha o seu jogo!*******")
print("**********************************")

print("(1) Forca (2) Adivinhação")

jogo = int(input("Qual jogo? "))

if (jogo == 1):
    print("Jogando forca")
    forca.jogar()
elif (jogo == 2):
    print("Jogando adivinhação")
    adivinhacao.jogar()
```

Podemos executar novamente o **jogos.py** e observar a sua saída:

```
**********************************
*******Escolha o seu jogo!*******
```

```
*******************************
(1) Forca (2) Adivinhação
Qual jogo?
```

Não aparecem mais as saídas dos jogos e sim a saída do próprio **jogos.py**. Os arquivos foram importados mas não foram executados, somente lidos. Eles só serão executados quando suas respectivas funções forem chamadas.

Com isso modularizamos o nosso código, temos um arquivo para cada jogo e um principal, que disponibiliza os jogos para o usuário. Podemos testar a execução do arquivo **jogos.py** pelo terminal também:

```
python3 jogos.py
```

E o jogo diretamente, será que conseguimos jogar? Se executarmos o arquivo **adivinhacao.py**, por exemplo:

```
python3 adivinhacao.py
```

Nada acontece! Ou seja, agora só conseguimos jogar os nossos jogos através do arquivo **jogos.py**. Vamos então tentar resolver isso no próximo vídeo, até lá!